An Empirical Investigation on Memes, Self-generation and Nurse Rostering

Ender Özcan

Yeditepe University, Department of Computer Engineering,
34755 Kadıköy/İstanbul, Turkey
eozcan@cse.yeditepe.edu.tr

Abstract. In this paper, an empirical study on self-generating multimeme memetic algorithms is presented. A set of well known benchmark functions is used during the experiments. Moreover, a heuristic template is introduced for solving timetabling problems. The heuristics designed based on this template can utilize a set of constraint-based hill climbers in a cooperative manner. Two such adaptive heuristics are described. Memetic algorithms utilizing each one as if a single hill climber are experimented on a set of random nurse rostering problem instances. Additionally, simple genetic algorithm and two self-generating multimeme memetic algorithms are compared to the proposed memetic algorithms and a previous study.

1 Introduction

Genetic Algorithms (GAs), introduced by J. Holland [27], are very promising for tackling complex problems [24]. Effectiveness of hill climbing methods in population based algorithms is underlined by many researchers [15, 38, 45, 46]. Memetic Algorithms (MAs) embody a class of algorithms that combine genetic algorithms and hill climbing methods. A *meme* represents a hill climbing method and its related parameters used within an MA. Ning et al. [39] concluded from their experiments that the meme choice in an MA influence the performance significantly. Krasnogor [31] extended the previous studies and suggested a *self-generating* (co-evolving) *multimeme* MA for solving problems in the existence of a set of hill climbers. Memes are evolved with the candidate solutions, providing a learning mechanism to fully utilize the provided hill climbers [32, 34].

In the first part of this study, the MA proposed by Krasnogor [33] is tested on a set of benchmark functions. The study aims to answer the following questions:

- Can the suggested learning mechanism discover useful hill climbers?
- Does a set of hill climbers generate a synergy to obtain the optimal solution?

In the second part of this study, MAs for solving a nurse rostering problem, introduced by Ozcan [41], are considered. Ozcan extended the study by Alkan et al. [5] and suggested templates designing a set of operators, including a self-adjusting violation-directed and constraint-based heuristics, named as VDHC, within MAs for solving timetabling problems. A new heuristic template for managing a set of constraint-

E. K. Burke, H. Rudová (Eds.): PATAT 2006, pp. 246–263. ISBN 80-210-3726-1.

based hill climbers is introduced in this paper. Two new instances based on this template are implemented and used as a single hill climber within MAs. Furthermore, two multimeme memetic algorithms (MMAs) are described. The performances of all the proposed algorithms, including the traditional genetic algorithm and the MA provided in [41] are compared.

2 Background

2.1 Benchmark Functions and Hill Climbing Methods

Benchmark functions with different features, well known among the evolutionary algorithm researchers, are utilized during the experiments (Table 1). F1-F11 are continuous, whereas F12-F14 are discrete benchmark functions. Detailed properties of each function can be found in the source references presented in Table 1. Benchmark functions include De Jong's test suite [17]. Only difference is that the noise component of the Quartic function is modified as described in [53].

Table 1. Benchmark functions used during the experiments: *lb* and *ub* indicate the lower and upper bound for each dimension, respectively, *opt* indicates the optimum

label	*function name*	*lb*	*ub*	*opt*	*source*
F1	Sphere	-5,12	5,12	0	[17]
F2	Rosenbrock	-2,048	2,048	0	[17]
F3	Step	-5,12	5,12	0	[17]
F4	Quartic *with noise*	-1,28	1,28	1	[53, 17]
F5	Foxhole	-65,536	65,536	0	[17]
F6	Rastrigin	-5,12	5,12	0	[47]
F7	Schwefel	-500	500	0	[50]
F8	Griewangk	-600	600	0	[27]
F9	Ackley	-32,768	32,768	0	[1]
F10	Easom	-100	100	-1	[20]
F11	Schwefel's Double Sum	-65,536	65,536	0	[51]
F12	Royal Road	-	-	0	[37]
F13	Goldberg	-	-	0	[25, 26]
F14	Whitley	-	-	0	[54]

Eight memes are used in the experiments:

- Steepest Descent (MA0), [37]
- Next Descent (MA1), [37]
- Random Mutation Hill Climbing (MA2), [37]
- Davis's Bit Hill Climbing (MA3), [16]

The remaining four memes are derived from the first two memes. The bit flip operation in MA0 and MA1 is replaced by an AND operation with 0, yielding MA4 and MA6, respectively. Similarly, an OR operation with 1 is employed, yielding MA5 and MA7, respectively. Gray and binary encodings are used to represent candidate solutions during benchmark experiments for continuous and discrete functions, respectively. Due to the gray encoding, the memes MA4-MA7 represent *poor* hill climbers for almost all continuous benchmark functions.

2.2 Nurse Rostering Problem

Timetabling problems are real-world constraint optimization problems. Due to their NP complete nature [20], traditional approaches might fail to generate a solution for an instance. Timetabling problems can be represented in terms of a three-tuple <*V*, *D*, *C*>, where *V* is a finite set of *variables*, *D* is a finite set of *domains* of variables and *C* is a set of *constraints* to be satisfied:
$V=\{v_1, v_2, \ldots, v_M\}$, $D=\{d_1, \ldots, d_i, \ldots, d_M\}$, $C=\{c_1, c_2, \ldots, c_K\}$

Solving a timetabling problem instance requires a search for finding the best assignment for all variables that satisfy all the constraints. Thus, a candidate solution is defined by an assignment of values from the domain to the variables:
$V' = \{v_1 = v_1', ..., v_i = v_i', ..., v_M = v_M'\}$, where $v_i' \in d_i$ and $d_i \subseteq D_1$ x … x D_P, where $P \geq 1$
In all timetabling problems, there is at least one domain for each variable that is for time. A problem instance might require other resources to be scheduled as well. For example, a university course timetabling problem instance might require arrangement of classrooms for each course meeting, as well. Then the search will be performed within a domain that will be a Cartesian product of time and classroom sets.

A *nurse roster* is a timetable consisting of employee shift assignments and the rest days of nurses in a health-care institution. Some health-care institutions might be composed of several departments. A *departmental roster* is defined as a collection of the nurse rosters of all nurses working within the same department. Nurse Rostering Problems (NRPs) are timetabling problems that seek for satisfactory schedules to be generated for employees, employers, even customers. In a common NRP, a nurse can be assigned to a day or a night shift, or can stay off-duty. A variable represents the shift assignment of a nurse. In this paper, *event* and *daily shift* will be used to refer *variable*, interchangeably. A *group* of events indicates a subset of events in *V* and their assignments in a candidate solution.

In all timetabling problems, constraints are classified as *hard* or *soft*. Hard constraints must be satisfied, while soft constraints represent preferences that are highly preferred. Furthermore, there are six different constraint categories for practical timetabling: *edge constraints*, *exclusions*, *presets*, *ordering constraints*, *event-spread constraint* and *attribute constraints* (includes *capacity constraints*) [42]. Edge constraints are the most common constraints that represent pairs of variables to be scheduled without a clash. A timetabling problem reduces to graph coloring problem, if the instance requires only edge constraints to be satisfied [35]. Exclusions determine the members to be excluded from the domain of variables for each variable. Presets are

used to fix the assignment of a variable. Ordering constraints, as the name suggests, are used to define an ordering between a pair of variables based on the timeline. Event-spread constraints define how the events will be spread out in time. Attribute constraints deal with the restrictions that apply between the attributes of a variable and/or the attributes of its assignment. Numerous researchers deal with NRPs based on different types of constraints utilizing variety of approaches. A recent survey on nurse rostering can be found in [10].

Burke et al. [8] applied variable neighborhood search using a set of different perturbation methods and local search algorithms on randomly generated schedules. Chun et al. [13] modeled nurse rostering as constraint satisfaction problem and embedded it as a Rostering Engine into the Staff Rostering System for the Hong Kong Hospital Authority. Similarly, Li et al. [36] modeled nurse rostering as a weighted constraint satisfaction problem. Their algorithm consists of two phases. In the first phase, forward checking, variable ordering and compulsory backjumping are used, whereas in the second phase descend local search and tabu search are used. Ahmad et al. [1] proposed a population-less cooperative genetic algorithm and experimented on a 3-shift problem. Kawanaka et al. [30] attempted to meet absolute and desirable constraints fro obtaining optimal nurse schedules. Aickelin et al. proposed a co-evolutionary pyramidal GA and experimented an indirect representation and three different decoders within GA for solving NRP in [3], [4], respectively. Gendreau et al. [23] used TS to generate shifts of nurses at the Jewish General Hospital of Montreal. Berrada et al. [6] combined TS with multiobjective approach, prioritizing the objectives. Heuristic swaps working and rest days. Duenas et al. [19] applied interactive Sequential Multiobjective Problem Solving method in conjunction with a genetic algorithm to produce a weekly schedule of eight nurses. Burke et al. [7] compared steepest descent, traditional TS and its hybrid with two local search heuristics for solving nurse rostering problem in Belgian Hospitals.

Recently, research on timetabling started to move towards finding a good *hyper-heuristic* [11]; a heuristic for selecting a heuristic among a set of them to solve an optimization problem. Cowling et al. [14] introduced hyper-heuristics as an iterative search method which maintains a single candidate solution and a set of heuristics. A hyper-heuristic is a heuristic utilized to choose a lower level heuristics. Han et al. [29] compared different versions of hyper-heuristics based on GA that they were developed for solving trainer scheduling problem utilizing fourteen different lower level heuristics. Burke et al. [12] proposed a tabu-search based hyper-heuristic, demonstrating its success for solving a set of nurse rostering problems at a UK hospital.

2.3 Multimeme Algorithms

Memetic Algorithms (MAs) are population based hybrid algorithms that combine Genetic Algorithms and hill climbing [15, 38, 45, 46]. In MAs, a *chromosome* (*individual*) represents a candidate solution to a problem at hand. A *gene* is a subsection of a chromosome that encodes the value of a single parameter (*allele*). Generally, the search for an optimal solution starts with a randomly generated set of individuals, called *initial population*. Then at each evolutionary step (*generation*) a set of opera-

tors are applied to each individual in the population. First, *mates* are selected for performing *crossover*, an operator that exchanges genetic material between mates. While selecting the mates, better ones are preferred. After the crossover, a set of new individuals, called *offspring*, is generated. Offspring are then *mutated*. In MAs, hill climbing operator is applied to the individuals, right after the crossover, or the mutation or in both places. Even the initial generation can be hill climbed. Whenever the *termination criteria* are satisfied, evolution stops. The best individual in the last generation is the best candidate solution achieved. In this paper, all MAs utilize a hill climber after the initialization and mutation.

Using a set of hill climbers, different MAs can be generated and compared for solving a problem. As another possibility, all hill climbers can be combined under a heuristic that selects one hill climber at a time and applies it. Such a hyper-heuristic schedules a hill climber in a *deterministic* or a *non-deterministic* way. For example, a deterministic round-robin strategy schedules the next hill climber in a queue. A non-deterministic strategy schedules the next hill climber randomly. These approaches employ blind choices. More complex and smart hyper-heuristics can be designed by making use of a learning mechanism that gets a feedback from the previous choices to select the right hill climber at each step. Different types of hyper-heuristics are discussed in [11].

Multimeme Algorithms (MMAs) represent a subset of self generating (co-evolving) MAs [31-34]. An individual in a population carries a memetic material along with a genetic material. The materials are co-evolved. In an evolutionary cycle, the memes are inherited to the offspring from the parents using the Simple Inheritance Mechanism (SIM) [33] during the crossover. SIM favors the meme of a mate with a better fitness to be transmitted to the offspring. In the case of an equal quality, a meme is randomly selected from the mates. Furthermore, a meme is altered to a random value based on a probability, called *Innovation Rate* (IR) during the mutation. MMAs, based on the SIM strategy and the mutation, allow modification of the candidate solutions by learning in order to obtain improved ones. This mechanism is referred as Lamarckian learning mechanism [31, 40].

Using a similar notation as provided in [33], a *meme*, denoted by M*h*FB*b*I*n*W*t*, represents the hill climbing method (M), its acceptance strategy (FB), the maximum number of iterations (I), and which part of the configuration to apply the selected method (W). An individual uses its meme to decide the hill climbing method and the related components to use, after the mutation takes place. Previously, Ong et al. [40] conducted tests on three benchmark functions using two new methods that they proposed for selecting the appropriate meme within MAs. In this study, MMAs are extensively tested on a set of well known benchmark functions. Furthermore, MMAs are used to determine where to apply a hill climber and which hill climber to apply, self adaptively for solving a real-world nurse rostering problem.

Success rate, *s.r.*, indicates the ratio of successful runs, achieving the expected fitness to the total number of runs repeated. Comparisons of MAs are based on the average number of evaluations and the success rate. Additionally, *average evolutionary activity* is considered during the assessment of MMA experiments. *Evolutionary activity* of a meme at a given generation is the total number of appearance of itself within each population starting from the initial generation until the given generation.

Average evolutionary activity is obtained by taking an average of the evolutionary activity of a meme at each generation over the runs. The slope of the average evolutionary activity versus generation curve shows how much a meme is favored. The steeper the slope gets for a meme, the more it is favored.

3 Memetic Algorithms for Benchmarking

3.1 Experimental Setup

All runs are repeated 50 times. Pentium IV 2 GHz. machines with 256 MB RAM are used during the experiments. Chromosome length, l, is the product of dimensions and the number of bits used. All the related parameters are arbitrarily chosen with respect to l. The mutation rate is chosen as a factor of $1/l$. The rest of the common parameter settings, used during the experiments are presented in Table 2. Runs are terminated whenever the overall CPU time exceeds 600 sec., or an expected fitness is achieved. All MAs use a tournament mate selection strategy with a tour size two, one point crossover, bit-flip mutation and a trans-generational MA with a replacement strategy that keeps only two best individuals from the previous generation. The IR rate is fixed as 0.20 during all multimeme experiments. A single acceptance strategy that approves only improving moves and a single value for the maximum number of hill climbing steps are used; b={1} and n={l}. A hill climber is applied to the whole individual; t={whole}.

Table 2. Common parameter settings used during the benchmark function experiments

label	*dim.*	*no. of bits*	*chrom. length*	*pop. size*	*max. hc steps*
F1	10	30	300	60	600
F2	10	30	300	60	600
F3	10	30	300	60	600
F4	10	30	300	60	600
F5	2	30	60	20	120
F6	10	30	300	60	600
F7	10	30	300	60	600
F8	10	30	300	60	600
F9	10	30	300	60	600
F10	6	30	180	36	360
F11	10	30	300	60	600
F12	8	8	64	20	128
F13	30	3	90	20	180
F14	6	4	24	20	48

During the initial set of experiments, the benchmark functions are tested using each meme described in Section 2.1. Experiments are also performed using a traditional Genetic Algorithm for comparison. The second set of experiments is designed according to the results obtained from the initial one. The best meme and two poor memes are fed into a multimeme algorithm. In the last set of MMA experiments, eight memes are used. Four hill climbing methods; h={MA0, MA1, MA2, MA3} are embedded. Hill climbing is applied depending on the acceptance strategy; b={0, 1}. 0 indicates a rejection, so hill climbing is not applied. If the meme points to the acceptance strategy 1, then the related hill climbing operator is applied. Hence, effectively there are five different memes. For short notation, each meme is referred as GA, MA0-MA3.

3.2 Empirical Results for the Benchmark Functions

Performance comparison of genetic algorithm and memetic algorithms using different memes are presented in Fig. 1 for selected benchmark functions based on the average number of evaluations. For each experiment, related bar appears in the figure, only if all the runs yield the expected result. MA0 is the best meme choice for F4, F13 and F14. MA1 is the best meme choice for F6-F8. MA3 is the best meme choice for F2, F3, F5, F10, and F12. For functions F1, F9 and F11 genetic algorithm performs slightly better than the memetic algorithm with the meme MA1. MA2 and MA3 turn out to be the worst and the best meme, respectively, among MA0-MA3.

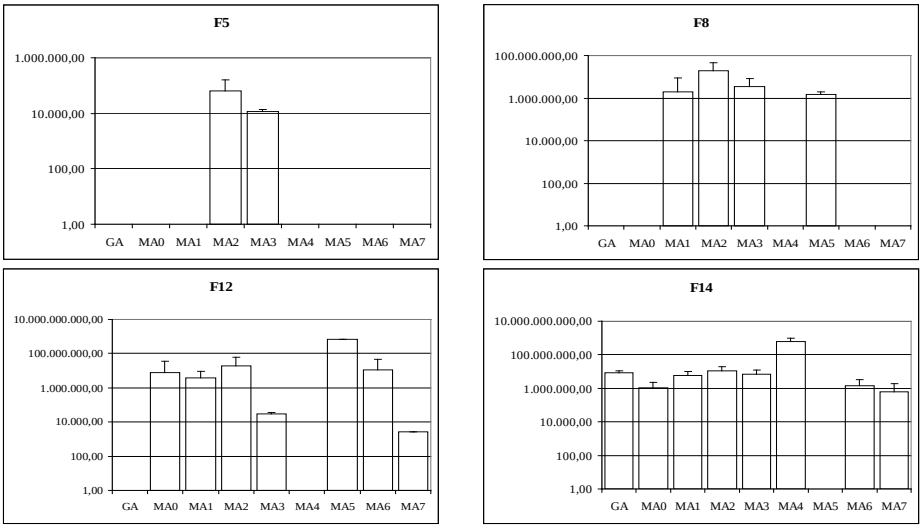


Fig. 1. Mean and the standard deviation of the number of evaluations per run, generated by each MA for a selected subset of benchmark functions

The average evolutionary activity versus generation plots generated during the second set of experiments show that the multimeme approach successfully identifies useful memes. The MMA chooses the best meme and applies it more than the rest of

the memes for all benchmark function, as illustrated in Fig. 2 for selected benchmark functions. The success rate for each benchmark function is 1.00. Any hill climber seems to attain the optimum fast for F1, F3 and F11.

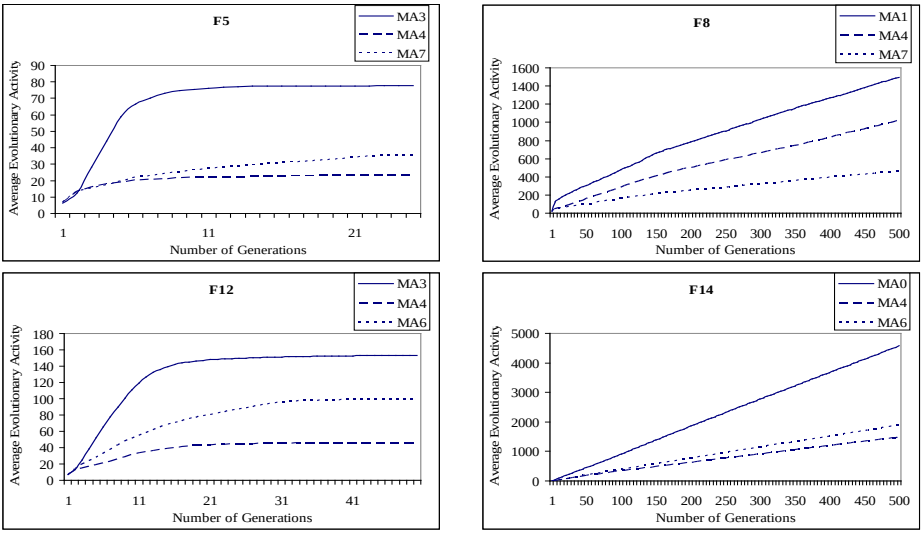
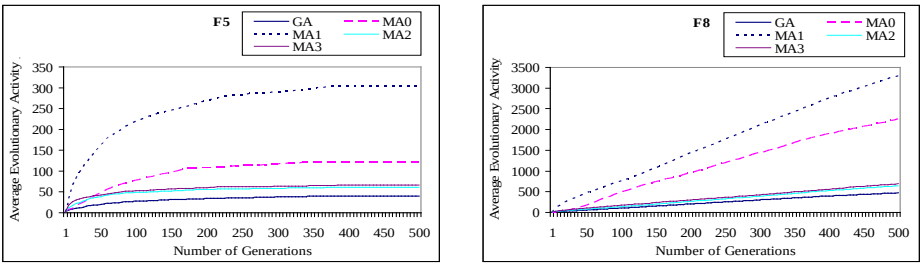


Fig. 2. Average evolutionary activity vs. generation plots of each meme utilized during the second set of experiments for a selected subset of benchmark functions

In the third and the last set of experiments, results similar to the previous one are obtained. The MMA can still identify the best meme or a meme that does not perform significantly better than the best meme for almost each benchmark function, as illustrated in Fig. 3 for selected benchmark functions. Furthermore, in all runs full success is achieved for all cases. Unfortunately, a synergy between hill climbers is not observed. Comparing the experimental results obtained using the MMA and the MA with the best meme for each benchmark indicate that the MA with the best meme is superior based on the average number of evaluations, except for F1, F3 and F11 (Table 3).



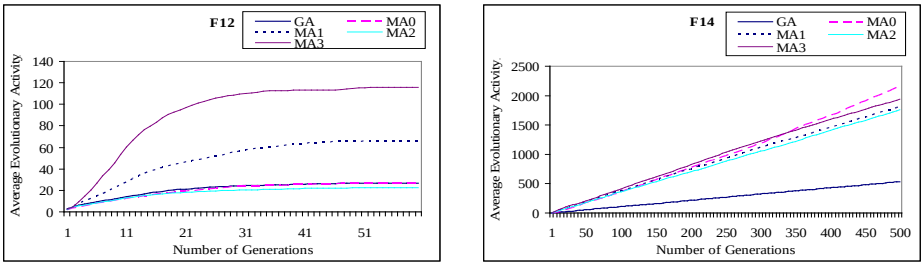


Fig. 3. Average evolutionary activity vs. generation plots of each meme utilized during the third set of experiments for a selected subset of benchmark functions

Table 3. Average number of evaluations and standard deviations generated by a Memetic Algorithm for each benchmark function: MA0-MA3 denotes the Memetic Algorithm using only the corresponding meme and MMA denotes the Multimeme Algorithm using all of them

label	*type*	*avr. no. of evals.*	*st.dev.*	*label*	*type*	*avr. no. of evals.*	*st.dev.*
F1	MMA	17,580	2,226	F8	MMA	5,215,787	9,658,230
	MA1	92,256	0		MA1	1,906,134	6,646,991
F2	MMA	23,605,004	24,364,979	F9	MMA	43,871	12,193
	MA3	8,455,507	3,803,504		MA1	180,783	12,647
F3	MMA	72,252	11,772	F10	MMA	3,100,515	4,565,736
	MA3	82,769	16,512		MA3	1,340,811	988,971
F4	MMA	12,926,879	11,435,876	F11	MMA	17,580	2,226
	MA0	9,494,844	10,332,574		MA1	36,060	0
F5	MMA	46,975	79,394	F12	MMA	31,297	14,961
	MA3	11,619	2,293		MA3	29,246	4,936
F6	MMA	553,306	231,124	F13	MMA	7,667,352	2,832,376
	MA1	525,398	262,055		MA0	4,348,896	1,617,951
F7	MMA	349,250	324,544	F14	MMA	3,674,932	2,623,300
	MA1	167,799	60,577		MA0	1,072,117	1,111,825

4 Memetic Algorithms for Nurse Rostering

4.1 Nurse Rostering Problem at the Memorial Hospital (NRPmh)

An analysis is performed on the Nurse Rostering Problem at the Memorial Hospital (NRPmh), located in İstanbul, Turkey. There are three types of daily shifts: *day*, *night* and *off-duty*. The timetable size is known in advance. Although a biweekly schedule

is preferred, the hospital authorities produce a weekly schedule manually, in order to simplify the timetabling process. Since the preferences of nurses are essential and might change in time, schedules are acyclic.

The hospital consists of three departments. Cross duty between the departments does not occur frequently. Hence, each nurse can be considered to be independent belonging to a specific department. Nurses are categorized into three ranks according to their experiences. Ranks {0, 1, 2} indicate the level of experience from lowest to highest. There are not many experienced nurses with rank 2, but there is at least one such nurse at each department. The constraints of this problem include;

Excludes:

– Exclude Night Shifts Constraint (ENC): Night shifts can not be assigned to an experienced nurse with rank 2.

Event-spread constraints:

– Off-duty Constraint (RDC): Nurses can define at most 4 rest day preferences.
– Shift Constraint (SHC): At a department, during each shift there must be at least one nurse.
– Successive Night Shifts Constraint (SNC): A nurse can not be assigned to more than two successive night shifts.
– Successive Day Shifts Constraint (SDC): A nurse can not be assigned to more than three successive day shifts.
– Successive Shifts Constraint (SSC): A nurse can not be assigned to two successive shifts. A day shift in one day and a night shift in the following day are considered as successive shifts.
– On-duty Constraint (ODC): Each nurse can not be assigned less than eight shifts per two weeks.

RDC is considered as a soft constraint, while the rest are hard constraints.

4.2 Constraint-based Violation-directed Heuristics

Ozcan [41] proposed a violation directed hierarchical hill climbing (VDHC) heuristic template to be used within MAs for solving timetabling problems and implemented an instance for solving a real-world nurse rostering problem. Experimental results show that it is a promising operator. In this study, a violation type directed hill climbing (VTDHC) heuristic template is presented as illustrated in Fig. 4. The VTDHC supports adaptation and cooperation of operators. It is a more general template than the VDHC.

The VTDHC template is designed to organize a set of hill climbers where each one improves a corresponding constraint type in a given timetabling problem. A set of events among several ones is selected based on the violations. The mechanism for selecting those events is up to the user. The number of violations caused by each constraint type within the selected set is used as a guide to select a hill climber. Finally, the selected hill climber is applied onto the selected events to resolve the violations due to the related constraint type.

```
1. while (termination criteria are not satisfied) do
       a. Select a group (or groups) of events based on vio-
          lations
       b. Select a constraint type based on contribution of
          each constraint type within the selected group (or
          groups)
       c. Apply hill climbing  for the selected  constraint
          type (without considering the other constraints)
          within the selected group of events
2. end while
```

Fig. 4. Pseudo-code of the VTDHC

An *event arrangement* indicates a structured organization of events in a timetabling problem. An event arrangement will be referred as *arrangement* in short from this point forward. For example, in Fig. 5, an arrangement for the NRPmh is provided. It is possible to identify more than one arrangement of events for a timetabling problem. Arrangements can be categorized as *static*, *dynamic* and *mixed*. An arrangement is labeled as static, if the members in a group do not change during the search process. In static arrangements, events can be hierarchically organized. Variables are logically grouped either as partitions or overlapping subsets at each hierarchy level of an arrangement. Static arrangement(s) can be obtained by analyzing the timetabling problem instance at hand. For example, according to the Nurse Rostering Problem described in the previous section, a static arrangement of variables can be derived as illustrated in Fig. 5. There are four hierarchical levels within the arrangement: Hospital, Department, Nurse and Variable. Hospital is a group including all variables, while a group in the Nurse level is a partition, where each indicates the roster of a nurse for two weeks. In this study, the static arrangement of daily nurse shifts (events) is used as shown in Fig. 5. Dynamic arrangements are based on the structure of the timetable and the assignment of events. Hence, members of a group might change during the search for an optimal solution, as the assignments of events might also change. For example, all the events (nurse shifts) scheduled at each day in a timetable constitute a dynamic arrangement of events. Mixed arrangements are a combination of both static and dynamic arrangements. For example, events scheduled at each day in a specific department represent a mixed arrangement.

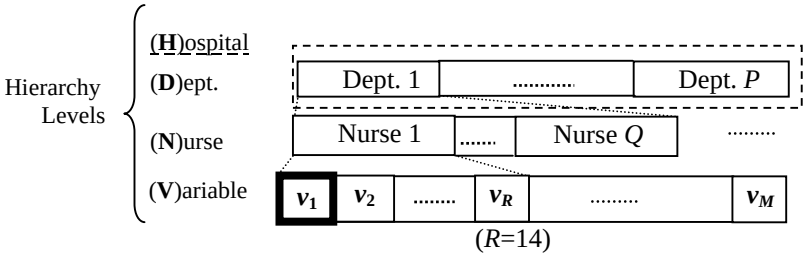


Fig. 5. Static arrangement of events (shifts) for NRPmh

Combining the arrangements and VTDHC yields the design of useful hyper-

heuristics. For example, VDHC represents a subset of VTDHC heuristics, using a static arrangement of events. It is an iterative heuristic that applies a selected hill climber to a selected group of daily shifts. The hill climber selection is constraint violation-driven and based on a predetermined arrangement. First, hierarchy levels of an arrangement to be used in the VDHC are decided. The top level is the starting level to operate on. As the candidate solution improves, it stays at a level. A selected hill climbing method is applied to a selected group of nurse shifts at a level, evaluating violations due to the each constraint type. The VDHC restricts the area of concern to the nurse shifts at one level down in the hierarchy in the case of a relapse and the same steps are repeated. It terminates whenever no improvement is provided in none of the levels or a maximum number of steps is exceeded.

A hill climber is selected using an implicit feedback from the evolutionary process, hence the VDHC is adaptive and in a way self-adjusting. During the traversal of an arrangement downwards in the hierarchy levels, the VDHC switches from individual level adaptation to component level adaptation [52]. In this study, two other hyper-heuristics are proposed based on the VTDHC template and used within MAs.

The VTDHC template can be extended and used for solving other multiobjective problems. Moreover, heuristics based on the VTDHC can be hybridized with other hyper-heuristics. In the current implementation, a single hill climber is designed for each objective. In the case of multiple hill climbers for each objective, the VTDHC instance can act as a decision mechanism for determining which objective to improve. Then, for the improvement of a selected objective, a traditional hyper-heuristic can be utilized to choose the hill climber to employ. This is a research direction beyond the scope of this paper.

4.3 MAs for Solving NRPmh

For solving the NRPmh described in Section 4.1, MAs are proposed. If there are T nurses in a hospital, then the total number of biweekly shifts to be arranged is $l=T*14$, where l is chromosome length. The search space size for finding the optimal schedule becomes immense; 3^l. The traditional approaches fail to obtain a solution, making MAs an appropriate choice. In all MAs, an allele in a chromosome represents a daily shift assignment of a nurse. Furthermore, each chromosome in the population is structured as illustrated in Fig. 5.

Seven hill climbing (HC) operators are designed to be used in the MAs: ENC_HC, RDC_HC, SHC_HC, SNC_HC, SDC_HC, SSC_HC, and ODC_HC. Each constraint based HC operator attempts to resolve the conflicts due to the related constraint for a given variable in an individual by random rescheduling. Details of the hill climbing operators can be found in [41]. In this study, five sets of experiments are performed. In each set, a different MA is used.

In the first set of experiments, a multimeme strategy for selecting which region to apply a selected hill climber is tested. The strategy also decides how many hill climbing steps should be used. Twelve different meme values are utilized. For all problem instances used during the experiments a single acceptance strategy is used;

b={1} and n changes from one problem instance to another. The values in n are fixed during the start of a run as {2l/4, 3l/4, l, 2l}. The values of t are {whole, department, nurse}. A meme acting as a scheduler determines whether a hill climber will be applied to the whole individual, or to a departmental roster or to a nurse roster. Then, a constrained type is determined to be improved for the group of shifts pointed by the meme. Using a tournament selection method with tour size of two, the constraint causing more violations within the group of shifts is favored among two randomly selected constraint types. Afterwards, the appropriate hill climber based on the selected constraint is applied to the group of shifts for a number of steps determined by the same meme. The MMA experiments using this operator are performed for three different IR values.

During the second set of experiments, hierarchical traversal of groups is reversed in the VDHC. The new hill climbing scheduler will be referred as rVDHC. Hill climbing starts from the bottom level; nurse level. As the candidate solution improves, the rVDHC stays at the nurse level. A selected hill climbing method is applied in the same way as the VDHC as described in Section 4.2. The rVDHC broadens the area of concern to nurse shifts in a whole department, which is one level up in the hierarchy, in the case of deterioration. Then the same steps are repeated. The termination criteria are the same as the VDHC.

In the third set of experiments a new scheduler is used. The worst nurse roster among a randomly selected two nurse rosters goes under a hill climbing process. This new scheduler is labeled as NHC. Notice that rVDHC and NHC are hyper-heuristics that are instances of VTDHC.

In the fourth set of experiments, a multimeme algorithm is implemented. MMA uses 7 memes; h={ ENC_HC, RDC_HC, SHC_HC, SNC_HC, SDC_HC, SSC_HC, ODC_HC }. All the rest of the parameters are fixed; b={1}, t={whole}, and n={2l}. Co-evolution determines which hill climber to apply. This version of the MMA is labeled as MMA7. The traditional GA is used during the last set of experiments in order to evaluate the role of hill climbers.

5 Nurse Rostering Experiments

5.1 Experimental Data and Common Settings

Runs are terminated whenever the overall CPU time exceeds 600 sec., or all the constraints are satisfied. The maximum number of hill climbing steps is fixed as 2l. All MAs for nurse rostering use ranking as a mate selection method, giving four times higher chance to the best individual to be selected than the worst one, one point crossover and a trans-generational memetic algorithm with a replacement strategy that keeps only two best individuals from the previous generation. The mutation operator is based on the traditional approach. A shift of a nurse is randomly perturbed with a mutation probability of 1/l. Based on the analysis of the NRPmh, six random problem instances are generated; rnd1-rnd6 and they are used during the experiments

[41]. The characteristics of the problem instances are summarized in Table 4. The data set is publicly available at http://cse.yeditepe.edu.tr/~eozcan/research/TTML.

Table 4. Experimental data set, where the number of departments and nurses are denoted as *ndep* and *nnur*, respectively. Percentage of nurses from each rank and average number of off-duty preferences of each nurse are denoted as *pnr* and *avrpr*, respectively.

Label	*ndep*	*nnur*	*pnr0*	*pnr1*	*pnr2*	*avrpr*
rnd1	3	21	0.42	0.32	0.28	1.95
rnd2	3	21	0.18	0.51	0.32	0.67
rnd3	3	21	0.28	0.42	0.32	2.19
rnd4	4	21	0.14	0.47	0.42	1.67
rnd5	4	21	0.19	0.46	0.37	2.33
rnd6	4	21	0.13	0.47	0.42	0.95

5.2 Empirical Results for the NRP Experiments

Detailed experimental results of the MA with the VDHC are presented in [41]. The results obtained from the first set of experiments indicate the viability of the MMA if used as a self adaptive method for selecting the region where to apply a hill climber. Yet, the MA with the VDHC performs better. Experiments are repeated for different values of IR around 0.20. The results are summarized in Table 5 for the experimental data. No IR value is significant. Considering the average success rates, all IR values yield almost the same performance. An interesting result of the first set of experiments is that MMA selects mostly a nurse roster and then applies a hill climber to it, as illustrated in Fig. 6 for IR=0.20. The rest of the experiments are performed on Pentium IV 3 GHz. machines with 2 GB RAM.

Table 5. MMA experiments using IR={0.15, 0.20, 0.25} with the random data set, where the first row denotes the success rate, the second row denotes the average number of generations per run for each IR value

IR	rnd1	rnd2	rnd3	rnd4	rnd5	rnd6
0.15	0.90	0.98	1.00	0.96	0.92	1.00
	1,145.96	217.74	77.54	697.28	667.58	234.08
0.20	0.94	0.98	1.00	0.94	0.94	1.00
	889.70	316.10	83.78	651.76	722.48	271.52
0.25	0.96	0.98	1.00	0.96	0.98	0.96
	921.12	317.62	92.20	750.18	371.34	422.90

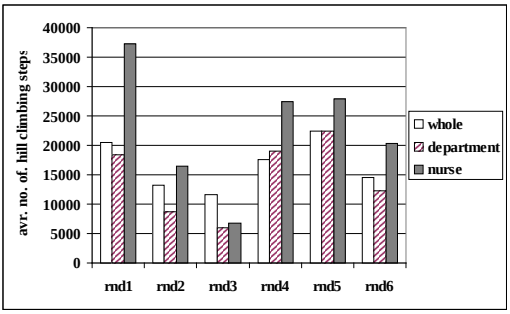


Fig. 6. Average number of hill climbing steps that are executed to improve the whole set of daily shifts, a departmental roster and a nurse roster for each problem instance during the first set of experiments, where IR=0.20

During the preceding sets of experiments, the MAs with *r*VDHC, NHC, the simple genetic algorithm and the MMA7 are tested on the problem instances. The success rate of each algorithm for each problem instance is presented in Table 6. Obviously, hill climbing boosts the performance GAs. Simple genetic algorithm turns out to be the worst algorithm for solving the problem instances. Almost; in none of the runs a violation free schedule is obtained. Empirical results yield the success of MAs with the following hill climbers from the best towards the worst: VDHC, *r*VDHC and NHC, respectively. The average performance of MMA7 is comparable to the performance of NHC. Results show that letting the multimeme algorithm to choose the region where to apply a constraint based hill climber based on a static hierarchical arrangement of events performs better than to let it to choose which meme to use for solving nurse rostering problem instances.

Table 6. The success rates of different algorithms for solving random problem instances

Label	VDHC	rVDHC	NHC	MMA7	Simple GA
rnd1	0.96	0.94	0.68	0.86	0.00
rnd2	1.00	0.98	0.88	0.96	0.04
rnd3	1.00	1.00	0.98	1.00	0.00
rnd4	0.98	0.94	0.28	0.18	0.00
rnd5	1.00	0.86	0.26	0.30	0.00
rnd6	1.00	1.00	0.68	0.50	0.00

6 Conclusions

Memetic algorithms, including the self-generating multimeme memetic algorithm proposed by Krasnogor [33] are investigated. Different MAs are experimented using a set of benchmark functions and nurse rostering problem instances, generated randomly by Ozcan [41] based on NRPmh. Some common empirical results are ob-

tained from both investigations. As expected, the performance of a genetic algorithm improves if a hill climbing operator is also utilized. Lamarckian learning mechanism employed by the MMAs yields appealing results for selecting a meme among a set of memes during the evolutionary process. Yet, the MAs with a good meme choice perform better. Different memes yield different performances. In the benchmark experiments, the MMAs identify the useful memes for all functions, but unfortunately, a synergy between hill climbers is not observed during the search. The average performance of the Davis's Bit Hill Climbing is the best on the benchmark functions.

The MAs are very promising approaches for tackling nurse rostering problems. Proposed heuristic template combined with a prior knowledge about a timetabling problem, such as a static arrangement, provides a promising guide for designing adaptive heuristics. The MAs, each containing such an instance as a single hill climber are compared to the MMAs, with different memetic materials. The empirical results indicate the success of the MA with VDHC [41] over the rest of the MAs presented in this paper. The VDHC using tournament selection provides a better cooperation among constraint-based memes. The hierarchical traversal over the groups based on a static arrangement during the hill climbing seems to work as well. Applying a constraint-based meme to a larger group of events first and then narrowing the area of concern generates better results than the reverse traversal. Still, the *r*VDHC shows potential.

Acknowledgement. This research is supported by TUBITAK (The Scientific and Technological Research Council of Turkey) under grant number 105E027.

References

1. Ackley, D.: An empirical study of bit vector function optimization. Genetic Algorithms and Simulated Annealing, (1987) 170-215
2. Ahmad, J., Yamamoto, M., and Ohuchi, A.: Evolutionary Algorithms for Nurse Scheduling Problem. Proc. of IEEE Congress on Evolutionary Computation (2000) 196-203.
3. Aickelin, U., and Bull, L.: On the Application of Hierarchical Coevolutionary Genetic Algorithms: Recombination and Evaluation Partners. JASS, 4(2) (2003) 2-17
4. Aickelin, U., and Dowsland, K.: An Indirect Genetic Algorithm for a Nurse Scheduling Problem. Computers & Operations Research, 31(5) (2003) 761-778
5. Alkan, A., and Ozcan, E.: Memetic Algorithms for Timetabling. Proc. of IEEE Congress on Evolutionary Computation (2003) 1796-1802
6. Berrada, I., Ferland, J., and Michelon, P.: A Multi-Objective Approach to Nurse Scheduling with both Hard and Soft Constraints. Socio-Economic Planning Science. vl. 30(1996)183-193
7. Burke, E.K., Cowling, P.I., De Causmaecker, P., and Vanden Berghe, G.: A Memetic Approach to the Nurse Rostering Problem, Applied Intelligence, vol 15 (2001) 199-214
8. Burke, E.K., De Causmaecker, P., Petrovic, S., Vanden Berghe G.: Variable Neighbourhood Search for Nurse Rostering Problems, in Metaheuristics: Computer Decision-Making (edited by M.G.C. Resende and J. P. de Sousa), Chapter 7, Kluwer (2003) 153-172

9. Burke, E.K., De Causmaecker, P., and Vanden Berghe, G.: A Hybrid Tabu Search Algorithm For the Nurse Rostering Problem, Proc. of the Second Asia-Pasific Conference on Simulated Evolution and Learning, vol. 1, Applications IV (1998) 187-194
10. Burke, E.K., De Causmaecker, P., and Vanden Berghe, G., Van Landeghem, H.: The State of the Art of Nurse Rostering, Journal of Scheduling, 7 (2004) 441-499
11. Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., and Schulenburg, S.: Handbook of metaheuristics, chapter 16, Hyper-heuristics: an emerging direction in modern search technology, Kluwer Academic Publisher (2003) 457-474
12. Burke, E., and Soubeiga, E.: Scheduling Nurses Using a Tabu-Search Hyper-heuristic, Proc. of the 1st MISTA, vol. 1 (2003) 197-218
13. Chun, A.H.W., Chan, S.H.C., Lam, G.P.S., Tsang, F.M.F., Wong, J., and Yeung, D.W.M.: Nurse Rostering at the Hospital Authority of Hong Kong, Proc. of 17th National Conference on AAAI and 12th Conference on IAAI (2000) 951-956
14. Cowling P., Kendall G., and Soubeiga E.: A Hyper-heuristic Approach to Scheduling a Sales Summit. Proceedings of In LNCS 2079, Practice and Theory of Automated Timetabling III : Third International Conference, PATAT 2000, Konstanz, Germany, selected papers (eds Burke E.K. and Erben W) (2000) 176-190
15. Davis, L.: The handbook of Genetic Algorithms, Van Nostrand Reingold, NY (1991)
16. Davis, L.: Bit Climbing, Representational Bias, and Test Suite Design, Proceeding of the 4th International conference on Genetic Algorithms (1991) 18-23
17. De Jong, K.: An analysis of the behaviour of a class of genetic adaptive systems. PhD thesis, University of Michigan (1975)
18. Downsland, K.: Nurse Scheduling with Tabu Search and Strategic Oscillation, European Journal of Operations Research. Vol. 106, 1198 (1998) 393-407
19. Duenas, A., Mort, N., Reeves, C., and Petrovic, D.: Handling Preferences Using Genetic Algorithms for the Nurse Scheduling Problem, Proc.of the 1st MISTA, vol.1(2003)180-196
20. Easom, E. E.: A survey of global optimization techniques. M. Eng. thesis, Univ. Louisville, Louisville, KY (1990)
21. Even, S., Itai, A., and Shamir, A.: On the Complexity of Timetable and Multicommodity Flow Problems, SIAM J. Comput., 5(4) (1976) 691-703
22. Fang, H.L. Genetic Algorithms in Timetabling and Scheduling, PhD thesis, Department of Artificial Intelligence, University of Edinburgh, Scotland (1994)
23. Gendrau, M., Buzon, I., Lapierre, S., Sadr, J., and Soriano, P.: A Tabu Search Heuristic to Generate Shift Schedules, Proc. of the 1st MISTA, vol.2 (2003) 526-528
24. Goldberg, D. E.: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading (MA) (1989)
25. Goldberg, D. E.: Genetic algorithms and Walsh functions: part I, a gentle introduction, Complex Systems (1989) 129-152
26. Goldberg, D. E.: Genetic algorithms and Walsh functions: part II, deception and its analysis, Complex Systems (1989) 153-171
27. Griewangk, A.O.: Generalized descent of global optimization. Journal of Optimization Theory and Applications (1981) 34: 11.39
28. Holland, J. H.: Adaptation in Natural and Artificial Systems, Univ. Mich. Press (1975)
29. Han, L., and Kendall, G.: Application of Genetic Algorithm Based Hyper-heuristic to Personnel Scheduling Problems, Proc. of the 1st MISTA, vol.2 (2003) 528-537
30. Kawanaka, H., Yamamoto, K., Yoshikawa, T., Shinogi, T., and Tsuruoka, S.: Genetic Algorithms with the Constraints for Nurse Scheduling Problem, Proc. of IEEE Congress on Evolutionary Computation (CEC), Seoul (2001) 1123-1130
31. Krasnogor, N.: Studies on the Theory and Design Space of Memetic Algorithms, PhD Thesis, University of the West of England, Bristol, United Kingdom (2002)

32. Krasnogor, N. and Smith, J.E.: Multimeme Algorithms for the Structure Prediction and Structure Comparison of Proteins. In Proc. of the Bird of a Feather Workshops, GECCO (2002) 42-44
33. Krasnogor, N. and Smith, J.E.: Emergence of Profitable Search strategies Based on a Simple Inheritance Mechanism. In Proc. of the Genetic and Evolutionary Computation Conference, GECCO (2001) 432-439.
34. Krasnogor, N. and Smith, J.E.: A Memetic Algorithm With Self-Adaptive Local Search: TSP as a case study. In Proc. of the Genetic and Evolutionary Computation Conference, GECCO (2000) 987-994.
35. Leighton, F. T.: A graph coloring algorithm for large scheduling problems. Journal of Research of the National Bureau of Standards, 84:489 (1979)
36. Li, H., Lim, A., and Rodrigues, B.: A Hybrid AI Approach for Nurse Rostering Problem, Proc. of the 2003 ACM Symposium on Applied Computing (2003) 730-735
37. Mitchell M., Forrest S.: Fitness Landscapes: Royal Road Functions, Handbook of Evolutionary Computation, Baeck T, Fogel D, Michalewicz Z (Ed.), Institute of Physics Publishing and Oxford Univers (1997)
38. Moscato, P., and Norman, M. G.: A Memetic Approach for the Traveling Salesman Problem Implementation of a Computational Ecology for Combinatorial Optimization on Message-Passing Systems, Parallel Computing and Transputer Applications (1992) 177-186
39. Ning, Z., Ong, Y. S., Wong, K. W. and Lim, M. H.: Choice of Memes In Memetic Algorithm, Proc. of the 2nd International Conference on Computational Intelligence, Robotics and Autonomous Systems (2003)
40. Ong, Y.S. and Keane, A.J.: Meta-Lamarckian Learning in Memetic Algorithms. IEEE Trans. Evolutionary Computation, vol. 8, no. 2 (2004) 99-110
41. Ozcan, E.: Memetic Algorithms for Nurse Rostering. Lecture Notes in Computer Science. Springer-Verlag, The 20th ISCIS (2005) 482-492
42. Ozcan, E.: Towards an XML based standard for Timetabling Problems: TTML, Multidisciplinary Scheduling: Theory and Applications, Springer Verlag (2005) 163 (24)
43. Ozcan, E., and Alkan, A.: Timetabling using a Steady State Genetic Algorithm, Proceedings of the 4th PATAT (2002) 104-107
44. Ozcan, E., Ersoy, E.: Final Exam Scheduler - FES, Proc. of 2005 IEEE Congress on Evolutionary Computation, vol. 2, (2005) 1356-1363
45. Ozcan, E., and Onbasioglu E.: Genetic Algorithms for Parallel Code Optimization, Proc. of 2004 IEEE Congress on Evolutionary Computation, vol. 2 (2004) 1775-1781
46. Radcliffe, N. J., and Surry, P.D.: Formal memetic algorithms, Evolutionary Computing: AISB Workshop, LNCS, vol. 865, Springer Verlag (1994) 1-16
47. Rastrigin L. A.: Extremal control systems. In Theoretical Foundations of Engineering Cybernetics Series. Moscow: Nauka, Russia. (1974)
48. Ross, P., Corne, D., and Fang, H-L.: Improving Evolutionary Timetabling with Delta Evaluation and Directed Mutation, Proc. of PPSN III (1994) 556-565
49. Ross, P., Corne, D., and Fang, H-L.: Fast Practical Evolutionary Timetabling, Proc. of AISB Workshop on Evolutionary Computation (1994) 250-263
50. Schwefel, H.-P.: Numerical optimization of computer models. Chichester: Wiley & Sons. (1981)
51. Schwefel, H. P.: Evolution and Optimum Seeking. John Wiley & Sons. (1995)
52. Smith, J. and Fogarty, T. C.:Operator and parameter adaptation in genetic algorithms. Soft Computing 1(2): 81-87 (1997)
53. Tasoulis D., Pavlidis N., Plagianakos V, Vrahatis M.: Parallel Differential Evolution. Proc. of 2004 IEEE Congress on Evolutionary Computation (2004) 2023-2029
54. Whitley, D.: Fundemental principles of deception in genetic search. In G.J.E. Rawlins (Ed.), Foundations of Genetic Algorithms. Morgan Kaufmann, San Matco, CA (1991)